ANL-7599

Argonne National Laboratory

C:READ, A Reentrant Routine to Convert EBCDIC Decimal Numbers to Hexadecimal

by

Conrad E. Thalmayer

The facilities of Argonne National Laboratory are owned by the United States Government. Under the terms of a contract (W-31-109-Eng-38) between the U. S. Atomic Energy Commission, Argonne Universities Association and The University of Chicago, the University employs the staff and operates the Laboratory in accordance with policies and programs formulated, approved and reviewed by the Association.

MEMBERS OF ARGONNE UNIVERSITIES ASSOCIATION

The University of Arizona
Carnegie-Mellon University
Case Western Reserve University
The University of Chicago
University of Cincinnati
Illinois Institute of Technology
University of Illinois
Indiana University
Iowa State University
The University of Iowa
Kansas State University
The University of Kansas
Loyola University
Marquette University
Michigan State University
The University of Michigan
University of Minnesota
University of Missouri
Northwestern University
University of Notre Dame
The Ohio State University
Ohio University
The Pennsylvania State University
Purdue University
Saint Louis University
Southern Illinois University
University of Texas
Washington University
Wayne State University
The University of Wisconsin

LEGAL NOTICE

This report was prepared as an account of Government sponsored work. Neither the United States, nor the Commission, nor any person acting on behalf of the Commission:

A. Makes any warranty or representation, expressed or implied, with respect to the accuracy, completeness, or usefulness of the information contained in this report, or that the use of any information, apparatus, method, or process disclosed in this report may not infringe privately owned rights; or

B. Assumes any liabilities with respect to the use of, or for damages resulting from the use of any information, apparatus, method, or process disclosed in this report.

As used in the above, "person acting on behalf of the Commission" includes any employee or contractor of the Commission, or employee of such contractor, to the extent that such employee or contractor of the Commission, or employee of such contractor prepares, disseminates, or provides access to, any information pursuant to his employment or contract with the Commission, or his employment with such contractor.

Printed in the United States of America
Available from
Clearinghouse for Federal Scientific and Technical Information
National Bureau of Standards, U. S. Department of Commerce
Springfield, Virginia 22151
Price: Printed Copy $3.00; Microfiche $0.65

ANL-7599
Mathematics and Computers

ARGONNE NATIONAL LABORATORY
9700 South Cass Avenue
Argonne, Illinois 60439

C:READ, A Reentrant Routine to Convert EBCDIC Decimal Numbers to Hexadecimal

by

Conrad E. Thalmayer

Chemistry Division

August 1969

TABLE OF CONTENTS

	Page
ABSTRACT	5
PREFACE	5
I. THE PROBLEM	5
II. GENERAL	6
III. EXTERNAL ORGANIZATION	7
IV. INTERNAL ORGANIZATION	8
SUMMARY	9
ACKNOWLEDGMENTS	9
APPENDIX A. Flow Charts	10
APPENDIX B. Listing	19

C:READ, A Reentrant Routine to Convert EBCDIC Decimal Numbers to Hexadecimal

by

Conrad E. Thalmayer

ABSTRACT

This report describes a reentrant, general-purpose routine for the Xerox Data Systems Sigma 5 or Sigma 7 computer with Floating-Point Option. C:READ converts EBCDIC decimal numbers in their normal input forms to hexadecimal numbers in the forms used in the computer. The report explains the need for the routine, describes its capabilities, presents all the information necessary for using it, and outlines its structure. The flow charts and listing are included.

PREFACE

This report describes a conversion routine for the Sigma 5 or Sigma 7 computer with Floating-Point Option. It is written in graded format, to be useful to readers of all levels of interest and sophistication. The general reader, for example, may profitably read the first one or two sections; the casual programmer will want to understand the second and third sections; only a programmer with special requirements will have need for the details of the fourth section, the flow charts and the program listing.

This routine is independent of the computer operating system. It was written in XDS SYMBOL in September 1968 and slightly amplified in April and June 1969.

I. THE PROBLEM

In Sigma computers, numbers are hexadecimal. Let us represent the hexadecimal digits, or "higits," as 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, and F and indicate a hexadecimal number by X'...'. Then, for example, the number X'1A' is equal to $(1 \cdot 16^1) + (10 \cdot 16^0) = 26$. Normally, numbers are of either word (8-higit) or doubleword (16-higit) length and either fixed point or floating point. A fixed-point number, necessarily integral, is equal to the sum of its higits, each successive higit leftward having been multiplied

by a successively higher power of 16; a floating-point number consists of a two-higit exponent followed by a 6 (or 14)-higit fraction.

Outside computers, numbers are (1) normally decimal, (2) of variable length, and (3) in various formats, e.g., signed or unsigned, with or without point, with or without exponent. Furthermore, (4) they are presented to the computer in EBCDIC (Extended Binary Coded Decimal Interchange Code); in this code, each character is represented by a two-digit number; e.g., '1' is represented as X'F1', and 'E' is X'C5'.

For input to the computer, a routine is necessary to convert numbers of the latter types into the former. The routine should be (1) rapid, (2) brief, (3) versatile enough to satisfy the needs of all programs using it, (4) amenable to convenient use in several ways, (5) capable of readily handling input from each of the usual devices, and (6) able to recognize all user errors and act appropriately. Most importantly, (7) the routine must be reentrant, i.e., while it is being used by a program of given priority, it must be interruptible by one of higher priority and subsequently resumable at the point of interruption; there should be no limit to the number of programs that might thus be sequentially interrupted while using the routine.

II. GENERAL

C:READ satisfies the above requirements. It accepts EBCDIC decimal numbers of various lengths in I, D, E, or F format, signed or unsigned, with or without leading or trailing blanks; it also accepts blank strings as zeros. It converts to short (8-higit) fixed-point or long (16-higit) floating-point numbers; the latter may be used as short numbers by ignoring the eight low-order higits.

The seven additional requirements listed above are abetted by the following: (1) This routine carries out only instructions pertinent to its specific task. It does not employ subroutines. (2) The various tasks use instructions of high commonality. (3) The number to be converted may be at any location and in a variety of forms. The value 5, for example, may appear as 5, 05, 5., +5, +05, +5., 5E0, .5E1, .5E+1, .5E 1, or +50.0E-1, to mention just a few possibilities. If the specified "number" is blank, it is interpreted as zero and the user is so informed. (4) The number may start at the "starting byte count" specified by the user, or it may be preceded by blanks. It may end at the specified "ending byte count" or at a blank; the latter is frequently convenient in that it enables the user to convert a subsequent number without change of specifications. (5) The routine itself recognizes relevant special characters, such as "End-of-Message," thus relieving the user from dealing with them. (6) The routine will reject a request if (a) a specification is illegal, (b) the number is too large or small, (c) the mantissa is too long, (d) a character is invalid, or (e) a sequence is

invalid, e.g., .+, 5+, E., or ++. The user is informed of the reason for the rejection. (7) The vital requirement of reentrancy is attained by carrying out all operations in the computer's registers. Upon interruption of a program, the contents of these registers and the address of the interruption are stored in that program's Program Description Table (PDT); upon return to the program, the registers are restored and execution is resumed at the interrupted instruction. This technique relieves the user of supplying some of his working space to the routine. Inasmuch as probably every real-time program will use this routine, this will result in a major saving of core space.

III. EXTERNAL ORGANIZATION

C:READ has two entry points: DECLREAD for conversion to floating point, and INTGREAD for conversion to fixed point. All exit points branch to the address given by the user in Register 0. The word address of the input field is given in R1, the starting byte count in that field in R2, and the ending byte count in R3. (Bytes are counted 0, 1, 2, ...) The end of the number may also be indicated prior to the ending byte count by a blank (except after D or E) or by the EOM character. As each byte (i.e., EBCDIC character) is treated, R2 is incremented by 1; it is therefore possible for the user to have several numbers, separated by blanks, in the one field; the user needs to set R2 at only his first entry to the routine--at each subsequent entry, R2 will already contain the correct starting byte count. If the user tries to read more numbers out of his field than it contains, or if he reads a portion of the field that is blank, the value 0 is returned and R6 is set to 0; reading an actual 0, or any other number, causes a 1 in R6. The routine returns the converted fixed-point number in R5 or the converted floating-point number in R4 and R5. Registers 7, 8, 9, 10, and 11 are also used; the remaining four registers are available to the user. This utilization of registers may be summarized as follows:

Input:

R0	User's return address
R1	Field Address
R2	Starting byte count
R3	Ending byte count

Output:

R2	Next byte count
R4,5	Number from DECLREAD
R5	Number from INTGREAD
R6	0 if pseudo 0 was read

This routine accepts input in a liberalized FORTRAN format. The given number may contain a mantissa, an exponent, or both; in the absence

of a mantissa, the value 1 is supplied. The beginning of the exponent field is indicated by either D or E. (Neither character carries any other implication.) The number may be preceded or followed by any number of blanks; a blank is also permitted immediately after D or E. The first character of the number, and the first character after D or E, may be + or -. If a mantissa is given, it may contain a decimal point. Leading zeros are permitted in both the mantissa and the exponent.

The largest number acceptable for INTGREAD is 2,147,483,640. The largest mantissa for DECLREAD is 72,057,594,037,927,935. The largest number for DECLREAD is 7.2370E75. The smallest number for DECLREAD is 5.3977E-79.

If the user's input is valid, the number is converted and the Condition Code is set to 0. If the input is not valid, the routine aborts to the address in R0 and the Condition Code is set to a value between 1 and 7, as follows:

1	Invalid format in mantissa
2	Invalid character in mantissa
3	Mantissa too long
4	Invalid format in exponent
5	Invalid character in exponent
6	Number too small
7	Number too large

IV. INTERNAL ORGANIZATION

C:READ has three parts: (1) mantissa evaluation, (2) exponent evaluation, and (3) number development. In Part 1, the first digit of the mantissa is evaluated, multiplied by 10, the second digit added, the RESULT multiplied by 10, etc. In Part 2, the same is done with the exponent. In Part 3, the exponent is reduced by the number of multiplications to the right of the decimal point in Part 1, and RESULT is multiplied (or divided) by 10 that many times. In the first two parts, each byte is interpreted completely before the next byte is examined.

In Part 1, the first byte is first checked for identity to blank, EOM, +, -, ., D, or E. Blank, in this situation, has no effect. EOM indicates the end of the number, now 0, and causes a transfer to Part 3. D or E causes a transfer to Part 2. Reading +, -, or . causes values in certain locations, actually registers, to be changed; these values will control the interpretation of subsequent bytes and are used in the final development. For example, reading - causes the assignments PASS = -1 and MSIGN = -1; PASS = -1 causes abortion if this is the end of the number; MSIGN = -1 prevents the acceptance of a subsequent + or - and, at the very end of the routine, causes complementation of RESULT.

Each byte, when it has been checked as just suggested, and not matched, is checked for identity to the digits. If this match fails, the routine aborts. If it succeeds, the value of the byte has been generated in the doubleword DIGIT. For the next stage, the routine diverges: if entry was at INTGREAD, RESULT is simply multiplied by 10 and DIGIT is added; if entry was at DECLREAD, as now shown by a flag set at entry, the same effect is accomplished by a complex series of operations on the doubleword RESULT. If the current digit was at some time preceded by ., DEC $<$ 1 and is now decremented by 1. Finally, in this part, MSIGN is set to +1 if no mantissa sign was read, and PASS is set to +1 to permit normal exit if this is the end of the number.

In Part 2, a series of preliminary tests is first made to check validity, and the mantissa is set to 1 if it did not exist. Each byte is then interpreted, somewhat as in the first part. The first byte may be +, -, blank, or a digit. The result of this part is the fixed-point quantity EXP.

In Part 3, the quantity DEC, if it is negative, is added to EXP. In the remainder of the routine, separate paths are followed for the two types of number. For DECLREAD, RESULT is first converted to floating point; it is then multiplied EXP number of times by 10 if EXP is positive, or divided if EXP is negative. For INTGREAD, RESULT is multiplied or divided by 10 the required number of times in fixed-point mode.

SUMMARY

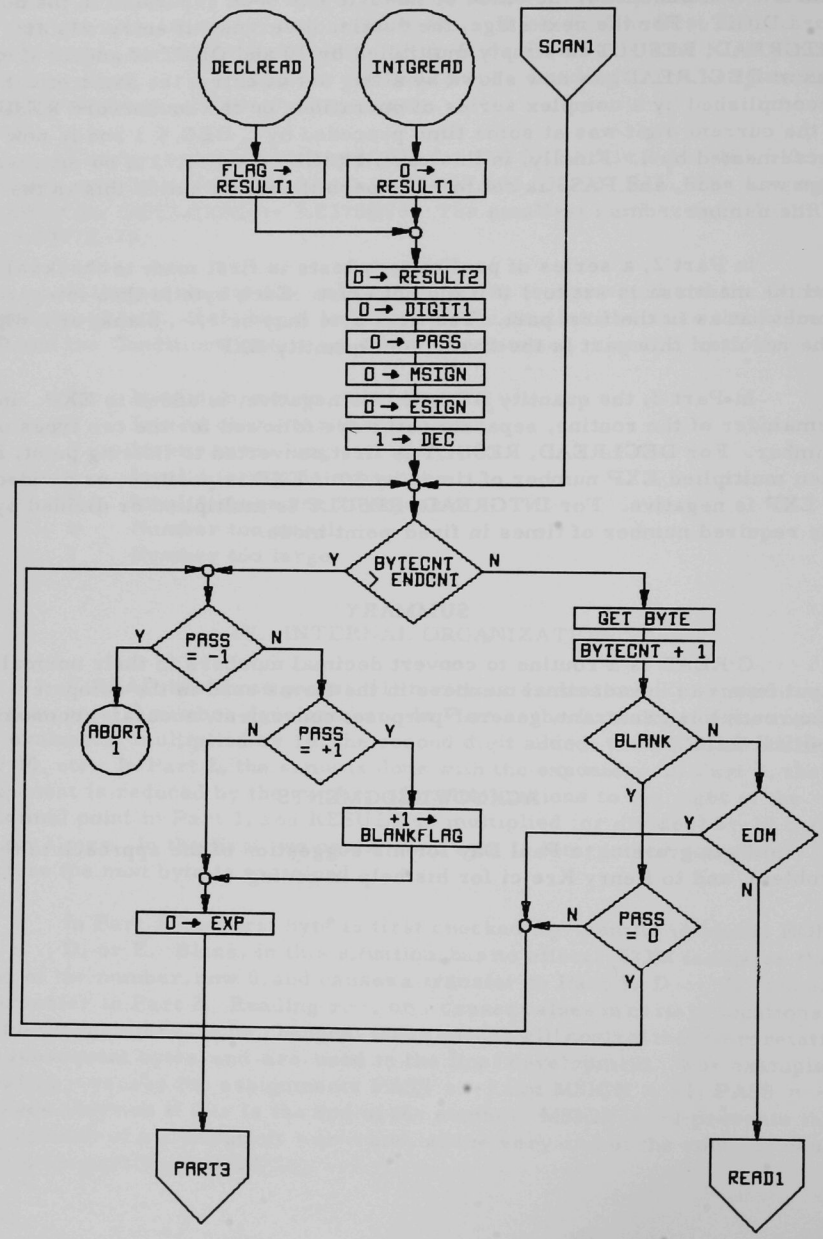
C:READ is a routine to convert decimal numbers in their normal input forms to hexadecimal numbers in the forms used in the computer. The routine is reentrant, general-purpose, convenient, accurate, economical, and fail-safe.

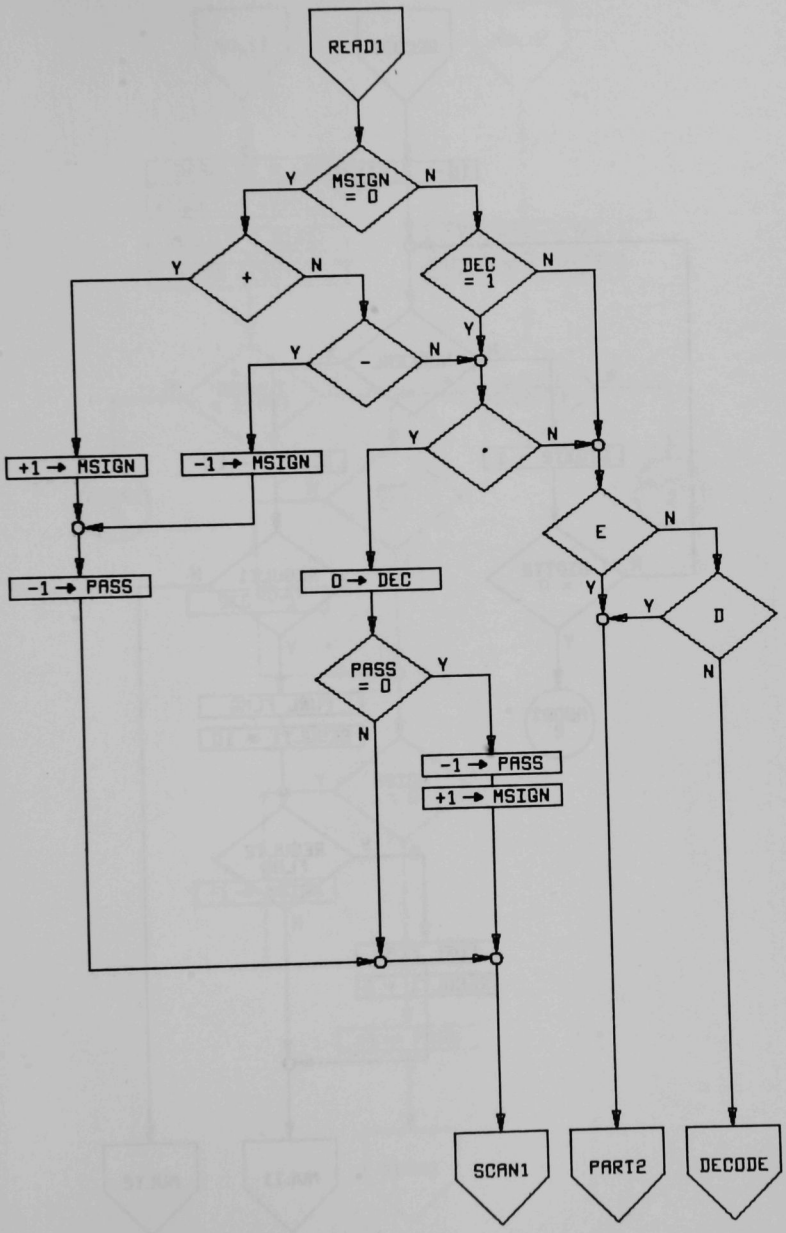
ACKNOWLEDGMENTS

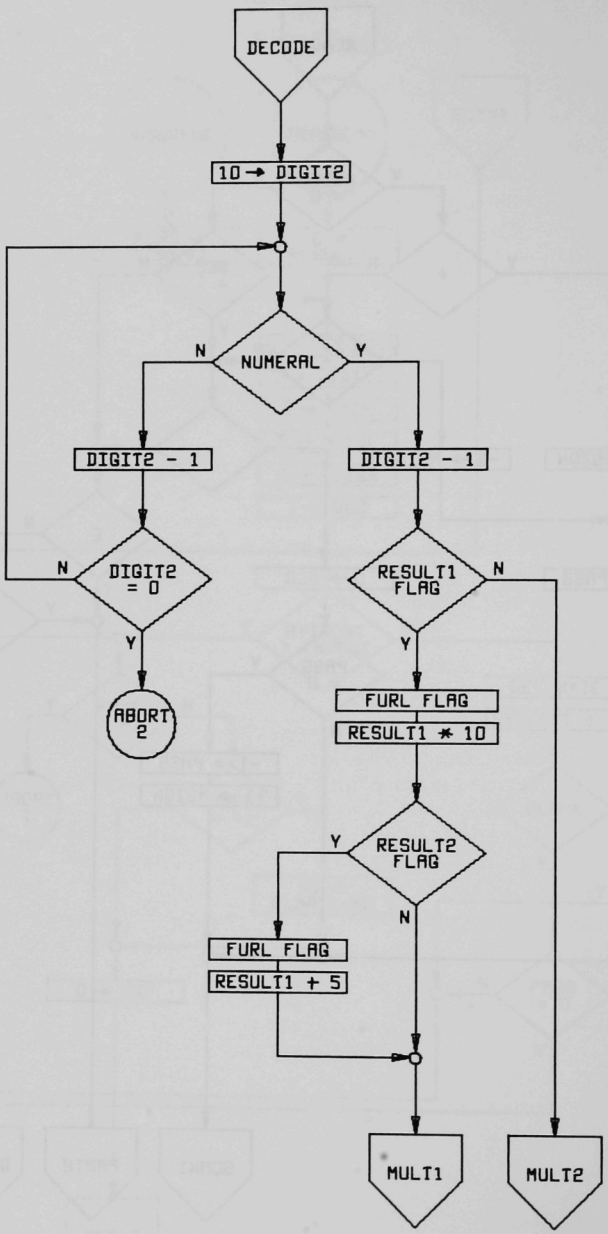
I am grateful to Paul Day for his suggestion of the approach to the problem, and to Henry Krejci for his help in coding.

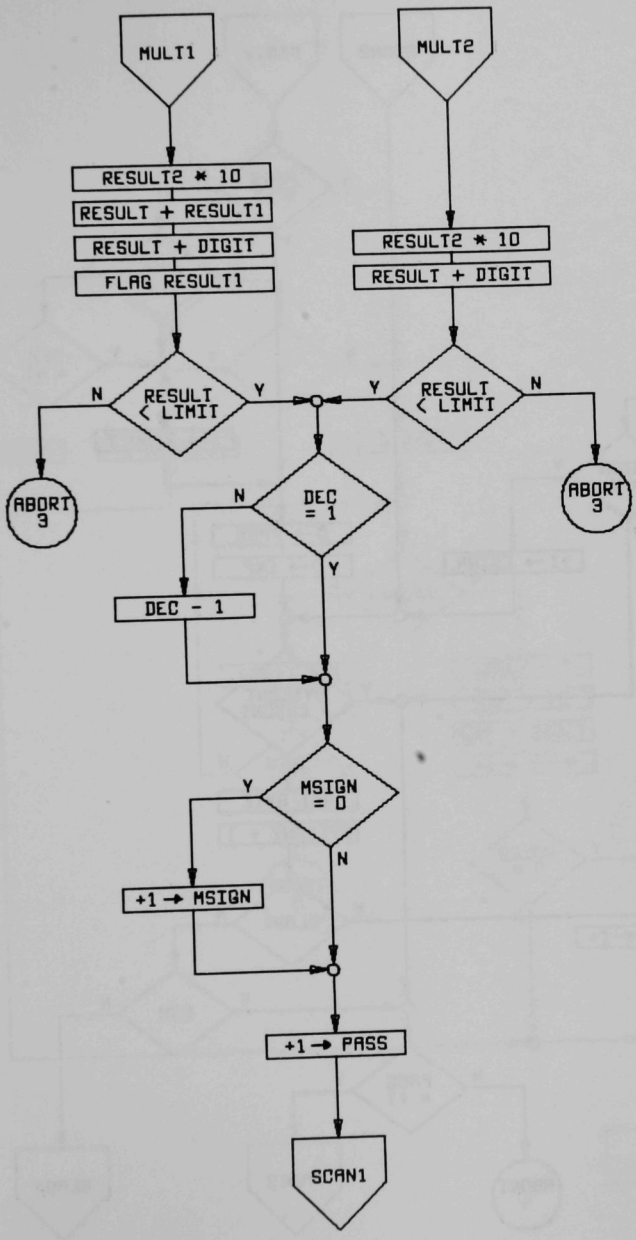
APPENDIX A

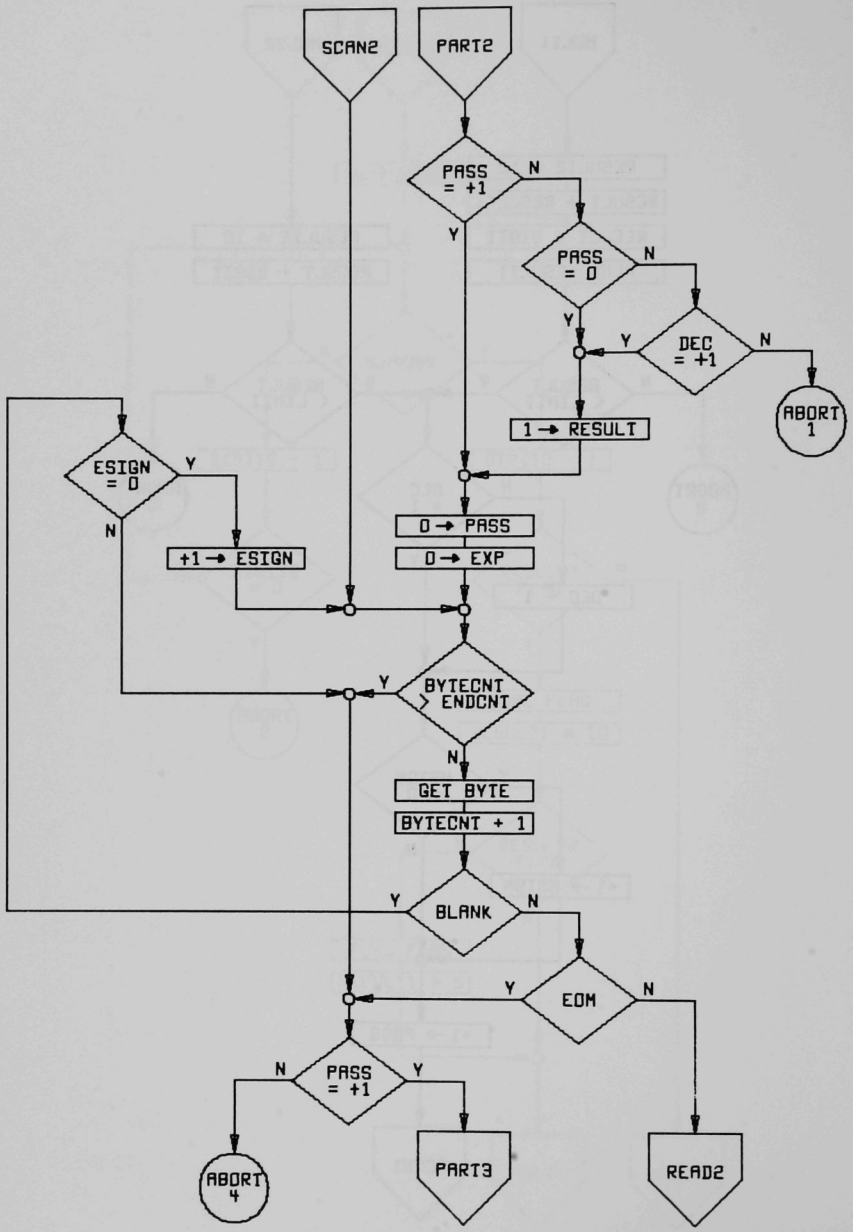
Flow Charts

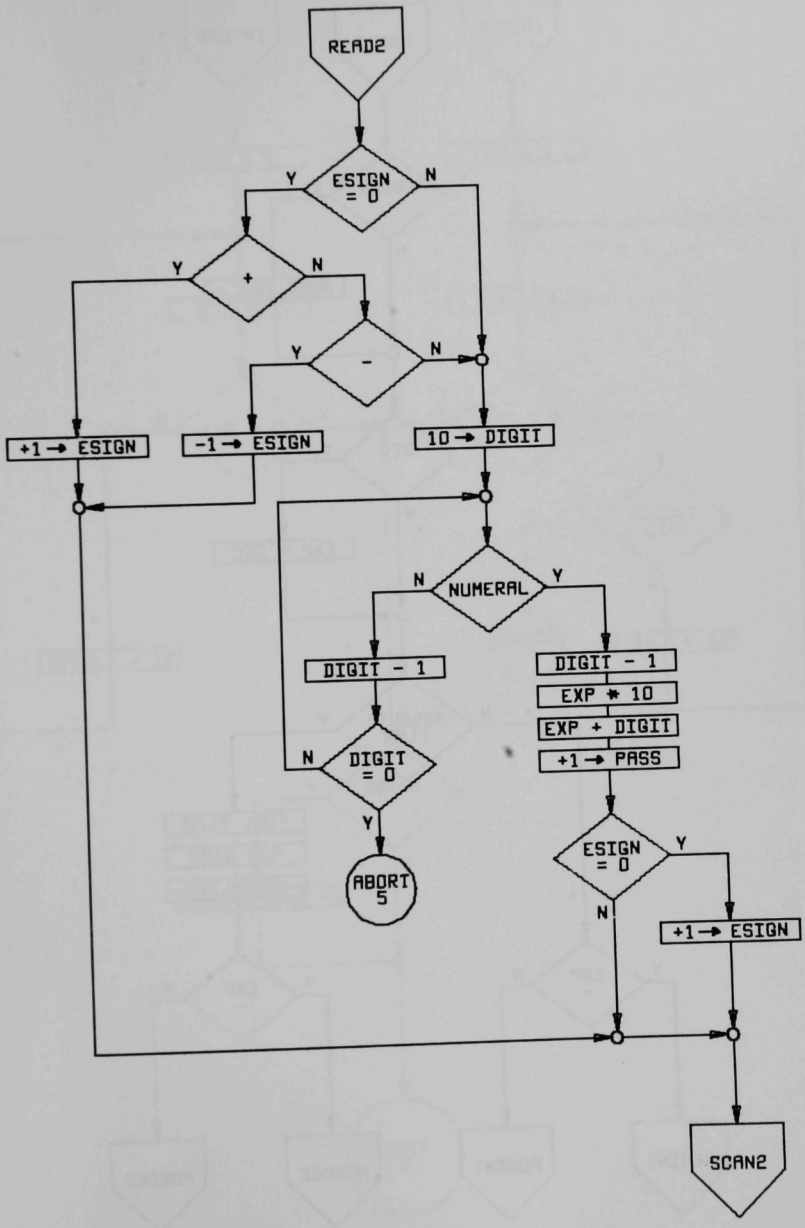


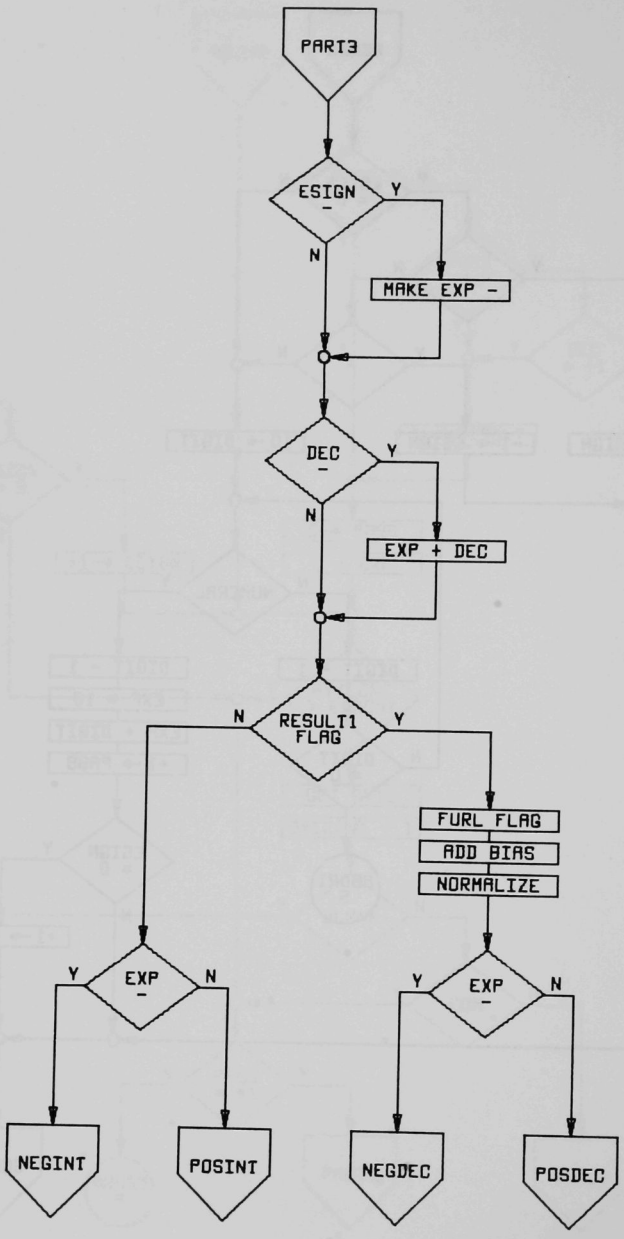

READ1
MSIGN = 0
Y
N
+
DEC = 1
-
.
E
D
+1 → MSIGN
-1 → MSIGN
-1 → PASS
0 → DEC
PASS = 0
-1 → PASS
+1 → MSIGN
SCAN1
PART2
DECODE

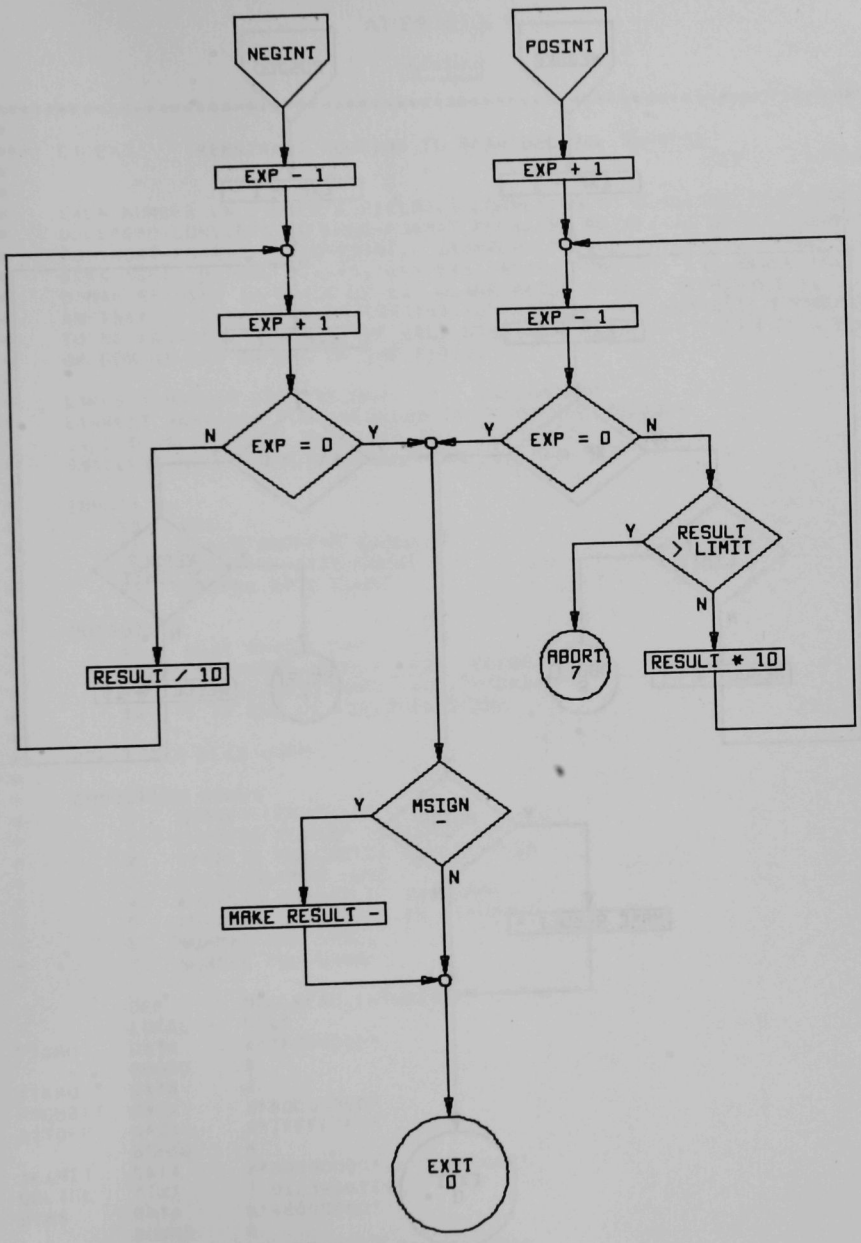

DECODE
10 → DIGIT2
NUMERAL
N
Y
DIGIT2 - 1
DIGIT2 - 1
DIGIT2 = 0
N
Y
ABORT 2
RESULT1 FLAG
N
Y
FURL FLAG
RESULT1 * 10
RESULT2 FLAG
Y
N
FURL FLAG
RESULT1 + 5
MULT1
MULT2

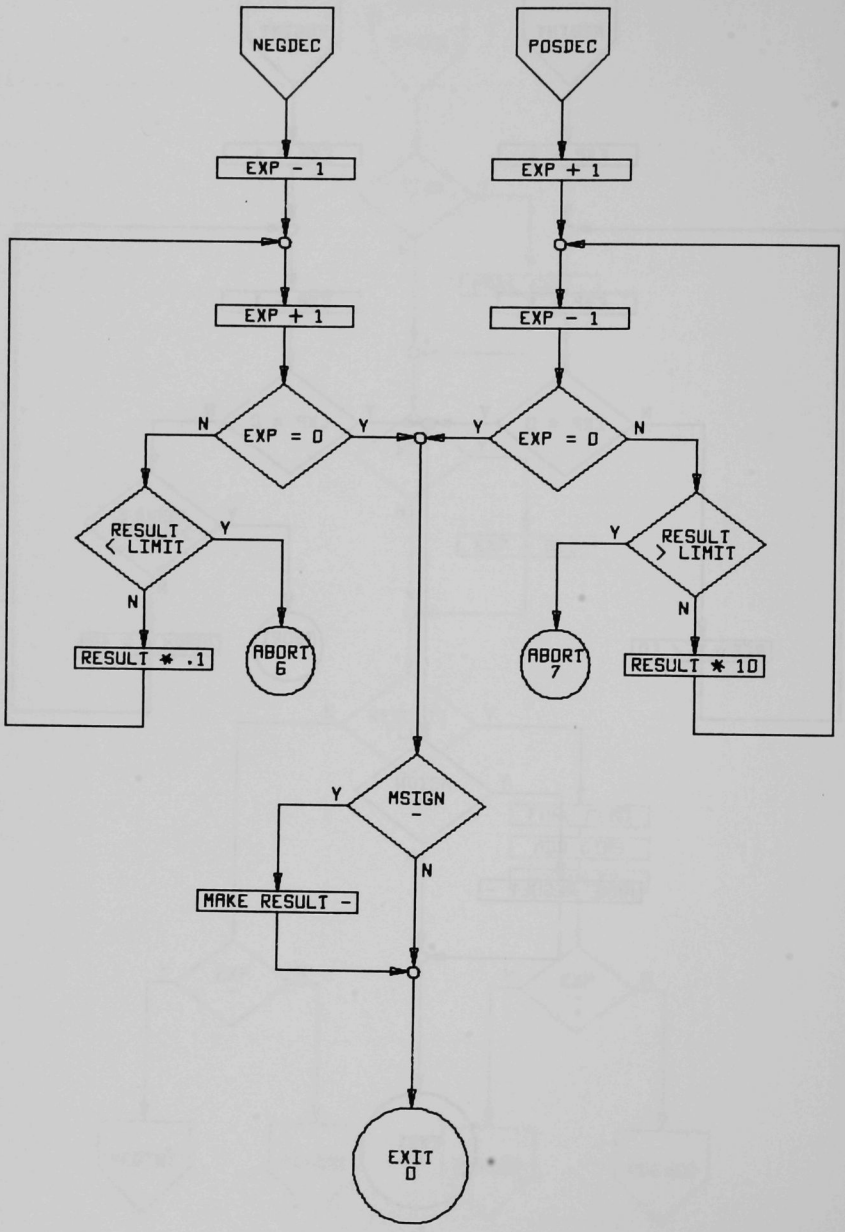

MULT1
MULT2
RESULT2 * 10
RESULT + RESULT1
RESULT + DIGIT
FLAG RESULT1
RESULT2 * 10
RESULT + DIGIT
N
RESULT < LIMIT
Y
Y
RESULT < LIMIT
N
ABORT 3
ABORT 3
N
DEC = 1
Y
DEC - 1
Y
MSIGN = 0
N
+1 → MSIGN
+1 → PASS
SCAN1


SCAN2
PART2
PASS = +1
N
Y
PASS = 0
N
Y
DEC = +1
Y
N
ABORT 1
1 → RESULT
ESIGN = 0
Y
N
+1 → ESIGN
0 → PASS
0 → EXP
BYTECNT > ENDCNT
Y
N
GET BYTE
BYTECNT + 1
BLANK
Y
N
EOM
Y
N
PASS = +1
N
Y
ABORT 4
PART3
READ2


READ2
ESIGN = 0
Y
N
+
Y
N
-
Y
N
+1 → ESIGN
-1 → ESIGN
10 → DIGIT
NUMERAL
N
Y
DIGIT - 1
DIGIT = 0
N
Y
ABORT 5
DIGIT - 1
EXP * 10
EXP + DIGIT
+1 → PASS
ESIGN = 0
Y
N
+1 → ESIGN
SCAN2


PART3
ESIGN
-
Y
N
MAKE EXP -
DEC
-
Y
N
EXP + DEC
RESULT1
FLAG
N
Y
FURL FLAG
ADD BIAS
NORMALIZE
Y
EXP
-
N
Y
EXP
-
N
NEGINT
POSINT
NEGDEC
POSDEC


NEGINT
POSINT
EXP - 1
EXP + 1
EXP + 1
EXP - 1
N
EXP = 0
Y
Y
EXP = 0
N
Y
RESULT
> LIMIT
N
RESULT / 10
ABORT
7
RESULT * 10
Y
MSIGN
-
N
MAKE RESULT -
EXIT
0


NEGDEC
POSDEC
EXP - 1
EXP + 1
EXP + 1
EXP - 1
N
EXP = 0
Y
Y
EXP = 0
N
RESULT < LIMIT
Y
N
RESULT * .1
ABORT 6
Y
RESULT > LIMIT
N
ABORT 7
RESULT * 10
Y
MSIGN -
N
MAKE RESULT -
EXIT 0

APPENDIX B

Listing

```
*************************************************************************
*
***  C:READ      REENTRANT ROUTINE TO READ DECIMAL NUMBERS
                                                          CET      6/10/69
*
*
*     EACH NUMBER IN A USER'S FIELD IS CONVERTED TO A HEXADECIMAL NUMBER.
*     DECLREAD CONVERTS TO LONG-FORMAT FLOATING POINT; INTGREAD CONVERTS
*     TO SHORT-FORMAT FIXED-POINT.  EXAMPLES OF VALID INPUT DECIMAL NUM-
*     BERS ARE:  5, 05, 5., +5, E5, E+5, +E+5, +05.E+5, AND BLANK (=0).
*     D MAY BE USED IN PLACE OF E.  BLANK AFTER E OR D SIGNIFIES +.
*     AN INVALID CHARACTER OR COMBINATION CAUSES THAT NUMBER'S CONVERSION
*     TO BE ABORTED.  THE END OF EACH USER NUMBER IS INDICATED BY A BLANK
*     OR EOM OR THE ENDING OF THE FIELD.
*
*     LARGEST NUMBER FOR INTGREAD IS    2147483640
*     LARGEST MANTISSA FOR DECLREAD IS 72057594037927935
*     LARGEST NUMBER FOR DECLREAD IS    7.2370E75
*     SMALLEST NUMBER FOR DECLREAD IS   5.3977E-79
*
*     INPUT:
*         R0    BAL
*         R1    FIELD ADDRESS (WORD)
*         R2    STARTING BYTE COUNT
*         R3    ENDING BYTE COUNT
*
*     OUTPUT:
*         R2    NEXT BYTE COUNT
*         R4,5  CONVERTED NUMBER FROM DECLREAD
*         R5    CONVERTED NUMBER FROM INTGREAD
*         R6    0 IF END OF FIELD IS BLANK
*
*     R7-11 ARE ALSO USED
*
*     CONDITION CODE:
*         0     NUMBER CONVERTED
*         1     INVALID FORMAT IN MANTISSA
*         2     INVALID CHARACTER IN MANTISSA
*         3     MANTISSA TOO LONG
*         4     INVALID FORMAT IN EXPONENT
*         5     INVALID CHARACTER IN EXPONENT
*         6     NUMBER TOO SMALL
*         7     NUMBER TOO LARGE
*
          DEF       DECLREAD,INTGREAD
          LOCAL     BIAS
FTABU     DATA      X'7F000000'
          BOUND     8
ITABU     DATA      0
HIGHBIT   DATA      X'80000000'
BITOFF    DATA      X'7FFFFFFF'
          BOUND     8
ILIMIT    DATA      X'00000000',X'0CCCCCCC'
DECTBL    TEXT      ' 0123456789'
BIAS      DATA      X'4E000000'
          BOUND     8
FPONE     DATA      X'40199999',X'9999999A'
FTEN      DATA      X'41A00000',X'00000000'
LLIMIT    DATA      X'00A00000',X'00000000'
ULIMIT    DATA      X'7F199999',X'99999999'
```

```
TEN       DATA    10
DECLREAD  LW,4    HIGHBIT       RESULT1; ENTRY FOR FLOATING-POINT
          B       CLEAR
INTGREAD  LI,4    0             RESULT1; ENTRY FOR FIXED-POINT
CLEAR     LI,5    0             RESULT2
          LI,6    0            H.O. DIGIT; PASS; DIGIT; BLANK FLAG
          LI,7    0             PASS; L.O. DIGIT; EXP
          LI,8    0             MSIGN
          LI,9    0             ESIGN
          LI,10   1             DEC
SCAN1     CW,3    2             IS FIELD ENDED
          BCS,1   EXIT1         END OF FIELD
          LB,11   *1,2          TAKE BYTE FOR INTERPRETATION
          AI,2    1             INCREMENT BYTE COUNT
          CI,11   ' '           IS BYTE BLANK
          BCS,3   NOBLANK       IF NOT BLANK
          CI,7    0             IS PASS=0
          BCR,3   SCAN1         IF PASS=0
EXIT1     CI,7    0
          BCS,1   *0           IF PASS=-1, ABORT (CC=1)
          BCR,2   SETEXP       IF PASS=0
          LI,6    1            BLANKFLAG
SETEXP    LI,7    0             SET EXP=0
          B       PART3
NOBLANK   CI,11   X'08'        CHECK FOR EOM
          BCR,3   EXIT1        IF EOM
          CI,8    0            IS MSIGN=0
          BCS,3   YESSIGN       IF SIGN EXISTS
          CI,11   '+'           IF NO PRIOR SIGN: CHECK FOR +
          BCS,3   NEGCHECK      IF NOT +
          LI,8    1             IF +
PASSNO    LI,7    -1            SIGN FOUND, NO DIGIT
          B       SCAN1         TAKE ANOTHER BYTE
NEGCHECK  CI,11   '-'           CHECK FOR -
          BCS,3   DECCHECK      IF NOT -
          LI,8    -1            IF -
          B       PASSNO
YESSIGN   CI,10   1             IS DEC=+1
          BCS,3   ECHECK        IF . EXISTS
DECCHECK  CI,11   '.'           CHECK FOR .
          BCS,3   ECHECK        IF NOT .
          LI,10   0             IF .
          CI,7    0             IS PSSS=0
          BCS,3   SCAN1         IF PASS NOT 0
          LI,7    -1            . FOUND, NO DIGIT
          LI,8    1             AS IF + FOUND
          B       SCAN1
ECHECK    CI,11   'E'           CHECK FOR E
          BCR,3   PART2         IF E
          CI,11   'D'          CHECK FOR D
          BCR,3   PART2        IF D
          LI,7    10            DIGIT INDEX
HUNT1     CB,11   DECTBL,7      CHECK FOR DIGIT
          BCR,3   DECODE1       IF DIGIT
          BDR,7   HUNT1         IF NOT DIGIT
          LCI     2             ERROR: INVALID MANTISSA CHARACTER
          B       *0            ABORT
DECODE1   AI,7    -1            DIGIT
          LW,11   4             HIGH-ORDER BITS OF RESULT
```

```
          BCR,1    INTG1       IF INTEGER
          AND,11   BITOFF      REMOVE FLAG
          MI,11    10          HIGH-ORDER PRODUCT
          CW,5     HIGHBIT     SIGN FLAG
          BCR,4    LOWMULT     IF ABSENT
          AND,5    BITOFF      REMOVE FLAG
          AI,11    5           A*8
LOWMULT   MI,4     10          LOW-ORDER PRODUCT IN R4,R5
          AW,4     11          TOTAL PRODUCT IN R4,R5
          AD,4     6           PRODUCT DOUBLEWORD + DIGIT DOUBLEWORD
          OR,4     HIGHBIT     REPLACE FLOATING-POINT FLAG
          CW,4     FTABU       IS HIGH BYTE EMPTY
          BCR,4    DECSHIFT    IF EMPTY
          LCI      3           ERROR: MANTISSA OVERFLOW
          B        *0          ABORT
INTG1     MI,4     10          PRODUCT IN R4,5
          AD,4     6           PRODUCT + DIGIT
          CD,4     ITABU       IS RESULT WITHIN LIMIT
          BCS,1    DECSHIFT    IF WITHIN
          LCI      3           ERROR: MANTISSA OVERFLOW
          B        *0          ABORT
DECSHIFT  CI,10    1           DOES . EXIST
          BCR,3    MSIGNCHK    IF NO .
          AI,10    -1          SHIFT .
MSIGNCHK  CI,8     0           IS MSIGN=0
          BCS,3    PASSYES     IF SIGN EXISTS
          LI,8     1           AS IF + FOUND
PASSYES   LI,7     1           DIGIT FOUND
          B        SCAN1       (END PART ONE)
PART2     CI,7     1           WAS DIGIT FOUND
          BCR,3    NOPASS      IF DIGIT FOUND
          CI,7     0           IS MANTISSA BLANK
          BCR,3    SETMAN      IF MANTISSA BLANK
          CI,10    1           WAS . FOUND
          BCS,1    *0          IF . FOUND, ABORT (CC=1)
SETMAN    LI,5     1           SET MANTISSA=1
NOPASS    LI,6     0           PASS
          LI,7     0           EXP
SCAN2     CW,3     2           IS FIELD ENDED
          BCR,1    GETBYTE     IF FIELD NOT ENDED
EXIT2     CI,6     1           WAS DIGIT FOUND
          BCR,3    PART3       IF DIGIT IN EXPONENT
          LCI      4           ERROR: INVALID EXPONENT FORMAT
          B        *0          ABORT
GETBYTE   LB,11    *1,2        TAKE BYTE FOR INTERPRETATION
          AI,2     1           INCREMENT BYTE COUNT
          CI,11    ' '         IS BYTE BLANK
          BCR,3    SUPPLUS     IF BLANK
          CI,11    X'08'       CHECK FOR EOM
          BCR,3    EXIT2       IF EOM
          CI,9     0            IS ESIGN=0
          BCS,3    DIGCHECK     IF SIGN EXISTS
          CI,11    '+'          CHECK FOR +
          BCS,3    MINCHECK     IF NOT +
          LI,9     1            SET ESIGN=+
          B        SCAN2

SUPPLUS   CI,9     0           DOES ESIGN EXIST
          BCS,3    EXIT2       IF ESIGN EXISTS
          LI,9     1           MAKE ESIGN +  (BLANK = SUPPRESSED +)
```

```
          B       SCAN2
MINCHECK  CI,11   '-'          CHECK FOR -
          BCS,3   DIGCHECK     IF NOT -
          LI,9    -1           SET ESIGN=-
          B       SCAN2
DIGCHECK  LI,6    10           DIGIT INDEX
HUNT2     CB,11   DECTBL,6     CHECK FOR DIGIT
          BCR,3   DECODE2      IF DIGIT
          BDR,6   HUNT2        IF NOT DIGIT
          LCI     5            ERROR: INVALID EXPONENT CHARACTER
          B       *0           ABORT
DECODE2   AI,6    -1           DIGIT
          MI,7    10           EXP * 10
          AW,7    6            EXP + DIGIT
          LI,6    1            SET PASS=1
          CI,9    0            IS ESIGN=0
          BCS,3   SCAN2        IF SIGN EXISTS
          LI,9    1            SET ESIGN=1
          B       SCAN2        (END OF PART TWO)
PART3     CI,9    0            IS ESIGN -
          BCR,1   DECNEG       IF NOT -
          LCW,7   7            MAKE EXP -
DECNEG    CI,10   0            IS DEC -
          BCR,1   DECORINT     IF NOT -
          AW,7    10           ADJUST FOR DECIMAL SHIFT
DECORINT  CW,4    HIGHBIT      FLOATING FLAG
          BCR,4   INTG2        IF INTEGER
          AND,4   BITOFF       REMOVE FLAG
          OR,4    BIAS         BIAS = 64+14
          SFL,4   13           NORMALIZE
          CI,7    0            IS EXP -
          BCR,1   POSEXP       IF NOT -
          AI,7    -1           FOR BIR
NEGBLD    BIR,7   LLCHK
          B       MANNEG
LLCHK     CD,4    LLIMIT       IS RESULT LESS THAN LOWER LIMIT
          BCR,1   DIVTEN       IF NOT
          LCI     6            ERROR: NUMBER TOO SMALL
          B       *0           ABORT
DIVTEN    FML,4   FPONE        DIVIDE RESULT BY TEN
          B       NEGBLD       RECYCLE
POSEXP    AI,7    1            FOR BDR
POSBLD    BDR,7   ULCHK        CYCLE
MANNEG    CI,8    0            IS MSIGN -
          BCR,1   DECLEND      IF NOT -
          LCD,4   4            MAKE NEGATIVE
DECLEND   LCI     0            NORMAL CC
          B       *0           NORMAL DECLREAD EXIT
ULCHK     CD,4    ULIMIT       IS RESULT GREATER THAN UPPER LIMIT
          BCR,2   MULTEN       IF NOT
          LCI     7            ERROR: NUMBER TOO LARGE
          B       *0           ABORT
MULTEN    FML,4   FTEN         MULTIPLY RESULT BY TEN
          B       POSBLD
INTG2     CI,7    0            IS EXP -
          BCR,1   POSEXPI      IF NOT -
          AI,7    -1           FOR BIR
NEGBLDI   BIR,7   DIVTENI
          B       INTNEG
```

```
DIVTENI   DW,5      TEN
          B         NEGBLDI
POSEXPI   AI,7      1                 FOR BDR
POSBLDI   BDR,7     ULCHKI
          B         INTNEG
ULCHKI    CD,4      ILIMIT            IS RESULT GREATER THAN LIMIT
          BCR,2     MULTENI           IF NOT
          LCI       7                 ERROR: NUMBER TOO LARGE
          B         *0                ABORT
MULTENI   MI,4      10
          B         POSBLDI
INTNEG    CI,8      0                 IS MSIGN -
          BCR,1     INTGEND          IF NOT -
          LCW,5     5                 MAKE RESULT -
INTGEND   LCI       0                 NORMAL CC
          B         *0                NORMAL INTGREAD EXIT
          END
```

ARGONNE NATIONAL LAB WEST
3 4444 00007861 8

B